Effective and Efficient Visualization and Analysis of Large-scale Data on Emerging Computer Architectures

Patrick S. McCormick, James Ahrens, CCS-1; Jeff Inman, HPC-1

Simulation and modeling play a fundamental role in many missions of the Department of Energy. These applications are producing data at ever-increasing rates that are overwhelming our capabilities to explore, hypothesize, document, and thus fully interpret the underlying details. These tasks will continue to become increasingly difficult as we make advancements towards petascale computing.

The computer architectures that support these computations are undergoing a revolutionary change as manufacturers transition to building chips that use an increasing number of processor cores. In addition, graphics hardware that was once designed entirely for rendering has rapidly evolved into a powerful general-purpose processor. While these trends have the ability to provide new capabilities and increase performance, they will place a significant strain on software development activities. The goal of our effort is to explore techniques that leverage the power provided by these emerging processor architectures to help scientists efficiently and effectively understand large amounts of information produced by computational simulations.

The fundamental challenge is how to provide scientists with a set of software tools that allow them to leverage the power of emerging computer architectures without requiring a detailed understanding of the available processors.

Driven by their wide use within the entertainment industry, graphics processors (GPUs) are among the most cost-effective, high-performance processors available today. Figure 1 provides a summary of the performance trends and prices of GPUs and CPUs as of August 2007. Similar trends are also evident in the design of other processor technologies. For example, IBM's Cell Broadband Engine is capable of 256 billion floating point operations per second [1]. Unfortunately, these processors are complex and can be very difficult to program efficiently.

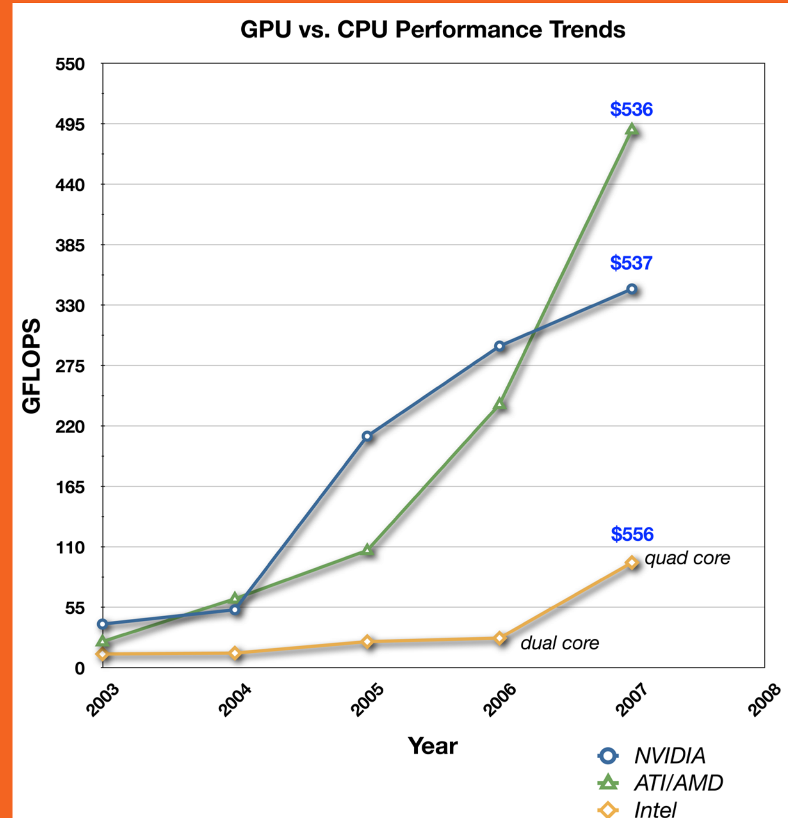

Fig. 1. Historical GPU versus CPU performance trends and current pricing for the most recent processors. Performance numbers courtesy of Mike Houston and John Owens. Pricing data from http://www.price-watch.com, August 2007.

To address this issue we have developed a parallel programming language targeted at supporting visualization, data analysis, and general-purpose computation. The Scout language allows visualization tasks, which are often represented by static user interfaces, to be directly expressed in programmatic form [2]. In addition, the supporting runtime and compiler infrastructure provides support for leveraging the power of multiple forms of parallelism and the potential for supporting computer systems designed using a collection of heterogeneous processors [3]. Figures 2 and 3 show example computations created using Scout.

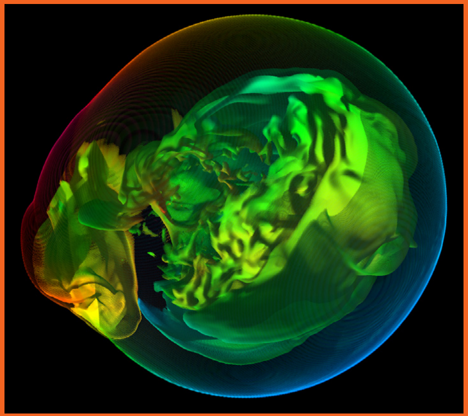

Fig. 2. Volume-rendered results of two selected entropy ranges colored by corresponding velocity magnitude. Both the entropy and velocity magnitude were computed directly using Scout. Performance was approximately 12 times faster than a 3.0 GHz Intel Xeon EM64T processor.

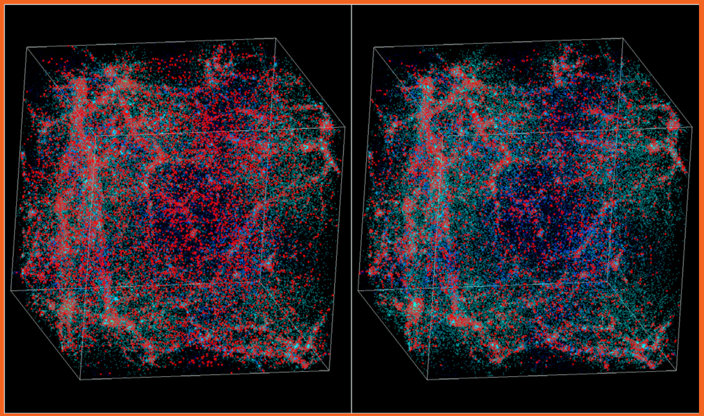

Fig. 3. A qualitative comparative visualization of the results produced by two different cosmology simulations codes studying the nonlinear structures in the dark and luminous matter that makes up the universe. The results were accelerated by leveraging the power of the graphics processor of a laptop computer. Using this approach we were able to compute, query, and render 32 million particles at 24 frames a second.

For further information contact Patrick S. McCormick at pat@lanl.gov.

[1] IBM Cell Broadband Engine Technology, http://www-03.ibm.com/technology/cell/.
[2] P.S. McCormick et al., *IEEE Visualization 2004*, October, 171-178 (2004).
[3] P.S. McCormick et al., *Parallel Computing*, **33**(10-11), 648-662 (2007).

Funding Acknowledgments
- Department of Energy, Office of Science, Office of Advanced Scientific Computing Research